ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240771

DataDetective: Dataset Watermarking for Leaker Identification in ML Training

Noa Wegerhoff*, Avishag Shapira, Yuval Elovici and Asaf Shabtai

Ben-Gurion University of the Negev

Abstract. Data owners (distributors) often share machine learning (ML) datasets with third-party collaborators (agents) for various purposes. While such collaborations can be mutually beneficial, they also introduce the risk of data leakage, i.e., the deliberate or accidental disclosure of sensitive ML datasets to unauthorized parties. Consequently, distributors may lose their intellectual property, experience reduced revenue, or violate data privacy regulations. In this paper, we propose a novel black-box dataset watermarking approach called DataDetective, which not only detects the unauthorized use of protected datasets but also identifies the agent responsible for the leakage. DataDetective, which leverages a backdoor technique, is composed of two processes: In the dataset watermarking process a unique *watermark signature* is embedded into each agent's version of the dataset, which embeds detectable, agent-specific behaviors in any model trained on the data. In the leaker identification process the watermark signature embedded in a suspected model is identified and compared to the signatures of all agents, to identify the leaking agent. Extensive evaluations on benchmark datasets in the computer vision domain demonstrate our method's effectiveness; DataDetective achieved a perfect leaker identification rate with just 1% of the data watermarked. Moreover, DataDetective maintains the model's performance with a negligible impact on model accuracy. By providing a verifiable and robust solution for leaker attribution, DataDetective enhances accountability in collaborative ML environments. For more details, the code is available at https://github.com/NoaWegerhoff/data-detective.

1 Introduction

Machine learning (ML) dataset sharing is a common practice in which data owners (hereafter distributors), such as governments, companies, and organizations, share their ML datasets with third-party collaborators (hereafter agents) to leverage the power of ML by model training [19]. These trained models can be used to enhance business intelligence [1], conduct data analytics [29], support research initiatives [34, 17], or develop commercial products [1].

The benefits of ML dataset sharing were demonstrated during the COVID-19 pandemic. Sharing large-scale medical datasets enabled researchers worldwide to rapidly develop ML models for crucial tasks such as predicting patient outcome and optimizing treatment strategies [34, 17]. This collaborative approach significantly accelerated the global response to the health crisis.

Shared ML datasets can be highly valuable to the data owners for two main reasons. First, these datasets can constitute valuable intellectual property (IP) when licensed or sold as commercial products, providing potential financial benefits to data owners [6]. Second, many datasets, include personally identifiable information (PII), such as names, national identification numbers, dates and places of birth, or biometric records [9].

While ML dataset sharing offers significant advantages, it also increases the risk of data leakage, i.e., the deliberate or unintentional distribution of sensitive or proprietary ML datasets to unauthorized parties [13], which may result in the protected ML dataset being used for unauthorized training of ML models.

The impact of ML dataset leakage is twofold: *(i)* Loss of ML dataset value. The creation of large datasets suitable for training ML models is a challenging process that requires significant resources and expertise. This process usually includes raw data collection, annotation, preprocessing, and validation [32, 26, 2, 6]. Unauthorized ML training on leaked proprietary (protected) ML datasets exposes the dataset's underlying patterns and knowledge, which poses financial risks and IP loss. *(ii)* Regulation violations. With the rise of regulations and ethical concerns surrounding data privacy and ML models (e.g., the GDPR [10] and CCPA [3]), agents may be constrained in their ability to disclose the data provided by the distributors [4]. Violation of these regulations may result in financial loss and/or reputation damage for both distributors and agents [13].

To realize the benefits of ML data sharing while mitigating these risks, and more specifically, to prevent protected data from being leaked and exploited by unauthorized entities for ML model training, distributors require a robust solution. This solution should enable them to not only detect leakage incidents but also identify the agents responsible for the leak, effectively tracing the source of a data breach when it occurs. To address the challenge of unauthorized model training detection, dataset watermarking has been proposed as a method to determine whether a specific dataset has been directly or indirectly incorporated into a machine learning model [27, 36, 21, 24, 22, 41]. However, these methods *cannot identify the source of the leak*, i.e., the agent responsible for enabling unauthorized access to the data.

We address this gap by introducing DataDetective, a novel dataset watermarking method that leverages backdoor attack techniques. Unlike existing methods that focus solely on ownership verification, DataDetective goes a step further by enabling the identification of the specific agent responsible for the leakage. This capability is crucial in scenarios where the distributor *only has black-box access* to the suspected ML model, meaning that they can query the model without direct access to the training data or the model's internals.

Our method generates a tailored version of the protected dataset

* Corresponding Author. Email: benamn@post.bgu.ac.il

Figure 1: **Dynamic Trigger Outputs**. The top row displays clean images, while the bottom row shows images watermarked with dynamic triggers. From left to right, the triggers used are a baseball, a Swiss flag, and a bee. These objects (triggers) were seamlessly inpainted into the images by a generator (DALLE 2), which adapts the triggers based on the scene in each image. The generated objects blend naturally and are tailored to fit the context of the image.

for each agent, embedding a traceable, unique behavioral signature into any model trained on that dataset. This involves two key processes: dataset watermarking and leaker identification.

In the **dataset watermarking process** (illustrated in Figure 2), each agent is assigned a unique *watermark signature*—a set of specific patterns (triggers) consistent across all agents, coupled with labels from the dataset (target classes) that are different for each agent. The agent's version of the dataset is created by embedding this unique watermark signature, where the triggers are embedded into a subset of samples, and the labels are altered to the corresponding target classes.

When trained on the watermarked dataset, a model learns to misclassify samples containing the trigger patterns as the coupled target class. We leverage this behavior in the **leaker identification process** to identify the embedded unique watermark signature, thereby pinpointing the corresponding agent as the source of the data leak.

We evaluated our method using the CIFAR-10 [20], CIFAR-100, and ImageNet [7] datasets from the computer vision domain (image classification task). To enhance the stealthiness and efficacy of our watermarking, we leverage generative AI models in the dataset watermarking process. This innovative technique ensures that the watermarks are inconspicuous (as shown in Figure 1), minimizing the risk of detection and removal by malicious actors. In our evaluation, DataDetective achieved a 100% success rate in identifying the leaking agent when as little as 1% of the data is watermarked. Furthermore, the performance of models trained with watermarked data remained robust, with just a small reduction in the accuracy (less than 5%).

The contributions of our work are as follows:

- We are the first to formally define and address the problem of identifying the entity responsible for leakage and unauthorized model training, proposing specific evaluation metrics and establishing a foundational approach for assessing solutions in this emerging research area.
- We propose DataDetective, a highly effective method that enhances dataset security by enabling precise ownership verification and leaking agent identification.
- We introduce a novel watermark signature scheme that enables efficient leaker identification, with the number of queries scaling logarithmically with the number of agents, making it scalable and practical.
- We introduce an inconspicuous method of watermarking datasets, which uses generative AI models, making it challenging for leakers to detect and remove the triggers, thereby enhancing the robustness of dataset protection.

2 Related Work

2.1 Machine Learning Backdoor Attacks

ML backdoors refer to techniques that embed concealed behaviors into trained ML models. Those hidden behaviors remain dormant until activated by a specific input trigger [23]. A common method for backdooring ML models is data poisoning [12, 38, 23]. In this approach, the adversary manipulates the training data of the target model and adds a predefined trigger to a selected set of data samples (e.g., a small patch in the case of images) and assigns a particular label to those samples [11]. The proportion of modified samples, known as the marking rate γ, is a key factor in the success of the attack [5]. As a result of this poisoning, the model learns to associate the trigger (backdoor pattern) with the intended target class and activates such behavior when encountering the trigger during inference [23]. In other words, when the poisoned model is presented with a sample containing the trigger, it is highly likely to misclassify it as the target class, even if the original sample belongs to a different class. BadNets [12] is a notable early backdoor attack that focuses on image classification tasks. Later studies suggested improving the attack by modifying the sample in such a way that it is indistinguishable from the original version to the human eye [5, 37]. Our method leverages the embedded behavior of a backdoor attack to identify the agent responsible for unauthorized training on a proprietary dataset while having only black-box model access to the model.

2.2 Data Leakage Detection

Data leakage detection (DLD) methods identify unauthorized confidential data disclosure and the responsible agent [13].

Papadimitriou et al. [31] introduced an agent guilt model to assess the probability of whether the data was disclosed by agents or an unauthorized party independently assembled it. The model evaluates agents' guilt probability when a distributor discovers leaked data in an unauthorized location. The existing methods examine the actual data samples, while we try to verify whether a suspected ML model was trained on the protected data without authorization and, if so, identify the agent guilty of leaking the protected dataset.

2.3 Dataset Provenance

A closely related field of study is dataset provenance (DP), which is also called dataset tracing. DP determines whether a data owner's proprietary data has been directly or indirectly integrated into a model trained by an unauthorized third party [27]. Existing dataset

provenance methods can be roughly divided into three main categories: dataset-level, user-data-level, and no-data adjustment methods.

Dataset-level adjustment methods, also known as dataset watermarking, protect a dataset by intentionally modifying (poisoning) data samples, thereby embedding a unique behavior into the model during training. The authors of radioactive data [36] proposed altering dataset images to induce a specific shift in the model's feature space representation, enabling dataset ownership verification via the statistical dependence between trained classifiers and thee shift direction. The DVBW method [21, 24] utilizes poison-only backdoor attacks for watermarking, with owners verifying suspicious models by performing hypothesis testing to confirm the embedded backdoor's existence.

In [22], the authors proposed using untargeted backdoor watermarks to mitigate security risks like increased adversarial vulnerability, causing models to exhibit abnormal behavior (e.g., higher errors, lower confidence) rather than specific misclassifications. A recent study [41] suggested a clean-label watermarking technique. The data owners first apply adversarial perturbations on selected samples to disable their useful features. Then, a preset backdoor trigger is applied.

While these techniques try to verify whether a dataset was used to train a model, our method's objective is broader. We aim to detect unauthorized use of a dataset and, more critically, determine the specific agent responsible for the leak, with only black-box access to the model.

User-data-level adjustment methods enable users to detect if their private data was used to train ML models. The authors of [40] exploited the fact that at inference time language processing models tend to assign higher ranks to low-frequency words when they appear in the same context they appeared in during training. The authors adjusted shadow models to train on the ranks of the user's target words in the output distributions as signals for inferring user-level membership. MIB [16] is an adjusted backdoor attack enabling user-level membership inference by stamping user data subsets with triggers and modifying labels to target classes. In [44], the authors presented the anti-neuron watermarking method, which performs linear color transformation as a unique private signature for each user. Verification is performed by a third-party arbitrator that marks a user's benign images with every possible signature. The arbitrator searches for the signature that produces the lowest loss in the suspected model. In the method suggested by [42], users' personal data is blended with "isotopes," which are unique inputs with features the model mistakenly considers predictive during training. Users can then statistically verify the presence of isotopes by querying the model.

User-data-level adjustment methods proposed in prior research primarily focus on determining whether the data of individual users was used in model training. These methods are user-centric, while our approach is agent-centric and works on a larger scale, aiming to identify which agent leaked an entire protected dataset, without needing access to the internal workings of the suspected model.

No data adjustment methods protect the IP of a dataset from unauthorized use and do not require any modification to the dataset or training process. The method proposed by [27] aims to prove the ownership of ML models and datasets in cases where a model has been stolen. The proposed method exploits the fact that models trained on different datasets will have different decision boundaries and uses the distance of protected data samples to the suspected model's decision boundary for IP verification. MeFA [26] extracts unique "fingerprints" (i.e., specific data samples that have similar impact on the prediction behavior across various models) of a protected dataset. Data ownership verification examines whether the unique fingerprints were part of a suspected model's training process.

No-data adjustment methods focus on proving ownership without modifying the data or model, basing their inferences on existing dataset characteristics. These methods require white-box access and assume some knowledge about the suspected model, while our method assumes only black-box access, making it more suitable for real-world scenarios.

Generally, DP methods are designed to solve a simple problem, aiming to protect just one dataset. In our approach, we consider the challenge of multiple distributed datasets, each with its unique markings, enabling us to detect unauthorized data usage in ML model training and effectively identify the agent responsible for leaking the dataset.

3 Proposed Method

3.1 Problem Statement

Our work focuses on computer vision classification problems and assumes the following setup. Let the $distributor$ be the owner of dataset D, a dataset to be protected by our method and shared with N different agents A_1, A_2, ... A_N. Agent A_l leaks the dataset to an unauthorized third party, which uses it to train a classification model f, and deploys it so users can access it via query. The $distributor$ suspects that f was trained on D without authorization and would like to identify the agent responsible for the data breach.

Our method employs a backdoor-based strategy enabling the $distributors$ to identify unauthorized use of their protected datasets and reveal the leaking agents. We assume the most realistic scenario where the querying approach to the suspected model is black-box with a probability vector output.

3.2 Guiding Principles for Solution

In designing a solution for leaker-identifying dataset watermarking, we base our approach on several key assumptions. Specifically, we assume that (1) there is no direct access to the architecture or training process of the suspected model, (2) there is no knowledge of the training data used, and (3) the number of agents involved in the data distribution might be very large.

With these assumptions in mind, we present the following principles that guided the development of our proposed solution:

- **Distinctiveness:** The watermarking and identification process should enable clear differentiation between the different agents. The method should ensure that models trained on watermarked datasets exhibit distinct behaviors, allowing precise identification of the responsible agent.
- **Minimal impact on performance:** The watermarking process should have a negligible impact on the performance of models trained on the watermarked datasets, preserving the original utility and accuracy of the data.
- **Inconspicuous:** The embedded watermark should remain inconspicuous, ensuring that the watermarked data does not raise suspicion or attract attention from third parties.
- **Scalability:** Ideally, the number of queries required in the leaking agent identification process should not scale directly with the number of agents the dataset has been distributed to. In practical terms, if the number of queries grows linearly ($O(\mathrm{N})$) as more agents are involved, several significant issues may arise:

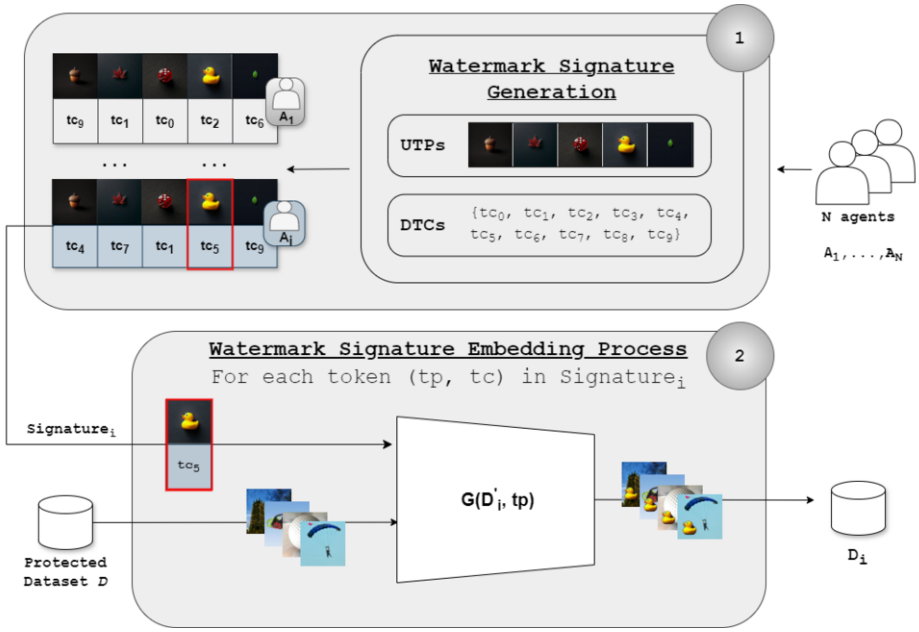

Figure 2: **Overview of the dataset watermarking process**. The process consists of two main stages: (1) **watermark signature generation**, where unique watermark signatures are created for each agent by a unique pairing of the UTPs with DTCs from the dataset, and (2) **watermark signature embedding**, where these signatures are embedded in the dataset by applying the triggers to selected data samples and labeling them with the corresponding target classes. The figure illustrates the flow from signature generation to embedding within the dataset.

- **Complexity and length:** An increase in the number of queries proportional to the number of agents can result in an identification process that is both overly complex and time-consuming. This scaling issue can make the process impractical in scenarios involving a large number of agents.
- **Risk of detection:** A large amount of queries may be perceived as an attempt to reverse engineer the model. Such querying patterns can raise suspicions and might prompt model owners to implement countermeasures that could detect and potentially block these queries [25].
- **Economic efficiency:** In cases where access to a suspected model is on a pay-per-query basis, the cost implications of a high number of queries can be substantial [6, 43]. Therefore, to mitigate any financial impact on the distributor, it is crucial to develop a method that efficiently identifies the leaker with a minimal number of queries.

3.3 Methodology Overview

Our method generates a tailored version of the protected dataset for each agent and relies on the ability to associate a single trigger pattern with different target classes. In other words, we can create a trigger that alters watermarked images so that they will be classified as target class tc_i, but we can also apply the same trigger to cause watermarked images to be identified as target class tc_j. We exploit this concept in order to assign unique watermark signatures to multiple agents, by embedding universal triggers that are common across all agents' datasets but associated with different target classes. This approach enables the creation of a one-to-one mapping between the unique combination of trigger pattern and target class pairs and a specific agent. While the trigger patterns remain constant and common across all agents, the distinctive combination of target classes assigned to each trigger pattern is used as a unique identifier for a particular agent.

DataDetective comprises two primary processes: In the **dataset watermarking process**, unique agent-identifying watermarks called *watermark signatures* are embedded in D. Each *signature* is a unique mapping between an agent, universal trigger patterns (UTPs), and class labels from the dataset (target classes), enabling the identification of the leaking agent when an unauthorized model is trained on the leaked dataset. In the **leaker identification process**, which is initiated upon suspicion that the protected dataset has been used without authorization to train a model f, the *watermark signature* embedded in model f is identified. This is done by identifying the target class associated with each trigger in the UTP set.

3.4 Dataset Watermarking Process

The dataset watermarking process focuses on creating a unique version of dataset D for each agent, denoted as D_i. This process involves four key elements:

- *Universal trigger patterns (UTPs)* - a set of k trigger patterns which are common across all agents' datasets.
- *Designated target classes (DTCs)* - a subset of the labels in the dataset that will be used as target classes for the triggers.
- *Token* - a pair consisting of (*trigger pattern (tp)*, *target class (tc)*).
- $signature_i$ - each agent A_i is assigned a unique signature:

$$signature_i = \{(tp_1, tc_1), \ldots, (tp_k, tc_k) \mid tp \in UTP, tc \in DTC\}$$

The dataset watermarking process is divided into two main stages: **watermark signature generation** and **watermark signature embedding**. Figure 2 presents a high-level flow of the dataset watermarking process.

3.4.1 Watermark Signature Generation

The watermark signatures are designed such that for any two agents A_i and A_j, where $i \neq j$, the corresponding signatures $signature_i$

and $signature_j$ differ by at least one token pair. In other words, there is at least one trigger pattern tp in which the coupled target class tc differs between $signature_i$ and $signature_j$. This ensures that each agent's watermark is unique and distinguishable from the others:

$$\begin{aligned} &\forall A_i, A_j \text{ where } i \neq j, \exists (tp, tc_p) \in \{\text{signature}_i\} \\ &\text{and } (tp, tc_r) \in \{\text{signature}_j\} \text{ such that } tc_p \neq tc_r \end{aligned} \tag{1}$$

The size of the signature, k (number of token pairs), should be chosen so that the number of unique watermark signatures is at least as large as the number of agents, N. We define $k = |UTP|$ (where $|UTP|$ is the size of the UTP set, i.e., the number of required triggers), by setting $|UTP| \geq \log_{|DTC|}(N)$, where $|DTC|$ is the number of available target classes in the dataset.

3.4.2 Watermark Signature Embedding

In the *watermark signature* embedding stage, D is modified to incorporate the agent-specific $signature_i$. Let $G(X, t)$ be a generator that blends trigger t into a set of images X. We start by assigning $D_i = D$, and then, through an iterative process, we embed $signature_i$ in D_i in the following manner: For each token $(tp, tc) \in \text{signature}_i$, a subset of D_i denoted as D'_i is selected. The trigger tp is applied to the subset D'_i using the generator G: $G(D'_i, tp)$, and the label (the target class) of D'_i is set to tc, producing a watermarked version of the subset denoted as $D'_{i,\text{modified}}$. The original subset D'_i is then replaced with $D'_{i,\text{modified}}$ in the dataset to create the watermarked dataset D_i. Thus, we update D_i as follows:

$$D_i = (D_i \setminus D'_i) \cup D'_{i,\text{modified}}$$

DataDetective's embedding process for a given signature (for an agent) is outlined in Algorithm 1.

Algorithm 1 Watermark Signature Embedding Process

Input: dataset D, $signature_i$, generator $G(X, t)$
$D_i \leftarrow D$
for $(tp, tc) \in signature_i$ **do**
    $D'_i \leftarrow$ Select a subset of D_i
    $D'_{i,\text{modified}} \leftarrow G(D'_i, tp)$
    Set $D'_{i,\text{modified}}$ labels to tc
    $D_i \leftarrow (D_i \setminus D'_i) \cup D'_{i,\text{modified}}$
return D_i

3.5 Leaker Identification Process

The leaker identification process is initiated when there is suspicion that the protected dataset has been used without authorization to train a model f. This process aims to identify the leaking agent A_l responsible for the unauthorized training by identifying the watermark signature $signature_l$ embedded in the model f. To identify $signature_l$, we apply a two-step statistical hypothesis-testing approach for each trigger pattern $tp \in UTP$. In the first step, the *Trigger Pattern Activation Test* (TPAT) determines whether the trigger pattern tp elicits a response in the suspected model f, indicating that f has been watermarked with tp. If the TPAT confirms the activation of tp, we proceed to the second step, the *Target Class Identification Test* (TCIT). The TCIT aims to identify the target class tc associated with tp during the training of f.

By sequentially applying these two steps for each $tp \in UTP$, we can identify the complete watermark signature $signature_l$ embedded in f. Once $signature_l$ has been identified, it can be compared to the signatures assigned to each agent in order to confidently pinpoint the leaking agent A_l responsible for the unauthorized training.

3.5.1 Step 1 - Trigger Pattern Activation Test (TPAT):

The TPAT determines if a specific trigger pattern tp activates a suspected model f, indicating that f was trained using the watermarked dataset containing tp. This is achieved by comparing f's classifications on benign test samples (X_{test}) and samples embedded with tp ($G(X_{test}, tp)$).

If f learned the trigger pattern tp during training, it should misclassify the trigger-embedded samples $G(X_{test}, tp)$, while correctly classifying the benign samples X_{test}. However, if f was trained on a benign training set, it should exhibit no significant difference in classifying trigger-embedded and benign samples. This behavior can be statistically tested by examining the difference between the distribution of the output labels for $f(X_{test})$ and $f(G(X_{test}, tp))$. In a case in which f is benign, we expect the distributions to be similar, and if f is watermarked, we expect the distributions to exhibit a statistical difference. We perform a hypothesis test to evaluate this behavior, where the null hypothesis (H_0) states that the label distributions of $f(X_{test})$ and $f(G(X_{test}, tp))$ are the same, while the alternative hypothesis (H_1) suggests a difference in these distributions.

3.5.2 Step 2 - Target Class Identification Test (TCIT):

If the TPAT confirms significant trigger pattern activation for tp, i.e, f is watermarked with tp, the TCIT is performed to identify the target class tc associated with the trigger pattern tp during f's training. To achieve this, we first query f using the benign test set embedded with trigger pattern tp ($G(X_{text}, tp)$). The model's predictions are recorded, and the most frequently predicted label, denoted as tc_s, is suspected as the target class associated with tp. For the suspected tc_s, we expect that the probability of the model f predicting tc_s for $G(X_{test}, tp)$ will be significantly higher than the probability of predicting tc_s for X_{test}. Therefore, we conduct a pairwise t-test between the probabilities $P((f(G(X_{test}, tp)) = tc_s)$ and $P(f(X_{test}) = tc_s)$. The null hypothesis (H0) states that these probabilities are similar, while the alternative hypothesis (H1) suggests that the probability $P((f(G(X_{test}, tp)) = tc_s)$ is significantly higher than the probability $P(f(X_{test}) = tc_s)$. A statistically significant result in the t-test, indicated by a p-value below the predefined threshold α, suggests that the suspected model f has been trained on a dataset marked with the token (tp, tc_s), thus revealing the watermark and implicating the corresponding agent A_l.

Together, the TPAT and TCIT enable the identification of the agent responsible for the data leak by identifying the trigger-target mappings embedded in the suspected model f.

3.6 Trigger Design and Implementation

Our method employs two distinct approaches for trigger design: static triggers and dynamic triggers. Each approach offers unique advantages in the context of dataset watermarking:

Static triggers are fixed, predetermined patterns or objects consistently applied across images. They are easy to implement and offer computational efficiency in embedding and detection processes.

Dynamic triggers are context-aware patterns or objects that are seamlessly integrated into images based on their content. As they adapt to each image's specific content, dynamic triggers preserve the visual quality and naturalness of the watermarked images and provide enhanced stealth.

The choice between static and dynamic triggers depends on the specific requirements of the watermarking task, balancing factors such as computational resources and the desired level of imperceptibility. The following subsections detail the implementation of each trigger type in our watermarking scheme.

3.6.1 Static Triggers

For static triggers, we follow an approach similar to BadNets [12]. We generate a set of trigger patterns $tp_1, tp_2, \ldots, tp_k$, where each tp is a fixed, static pattern (e.g., a simple shape like a duck or an apple). Let D'_i be a subset of images from dataset D_i; to embed a trigger pattern tp in D'_i, we apply the following operation:

$$G(D'_i, tp) = (1 - \alpha) \otimes x + \alpha \otimes tp | x \in D'_i \quad (2)$$

where α is a blending factor that controls the intensity of the trigger pattern tp, and $\otimes$ represents element-wise multiplication. Essentially, generator G overlays the trigger pattern tp onto each image x in the subset D' as a static overlay or sticker in the BadNets approach. Examples of blending static triggers using varying blending factors can be seen in Figure 3.

3.6.2 Dynamic Triggers

For dynamic triggers, we leverage generative AI models to seamlessly blend triggers into the images in a natural and contextual manner. Let D'_i be a subset of images from dataset D_i. We define a trigger pattern tp as a dynamic object or entity (e.g., a bee, Swiss flag) that can be blended into images in different ways. Examples of blending dynamic triggers can be seen in Figure 1. The dynamic trigger tp embedding procedure involves the following steps for each image $x \in D'_i$:

1. We use a pretrained large multimodal model (LMM), such as GPT-4 with vision [30], to identify the most suitable embedding locations and generate a mask x_m for embedding the trigger object tp in the image.
2. Then, we employ a generator G that utilizes a generative AI model trained for inpainting tasks. G receives the mask x_m and the trigger tp, and seamlessly blend tp into the region specified by x_m, naturally and contextually.

For simplicity, for each trigger pattern tp, we chose a subset of images that originally belonged to the same target class and embedded the trigger in these images.

4 Evaluation

4.1 Evaluation Setup

Models. We perform experiments on two widely-used classification models in our evaluation: ResNet-18 [14] and VGG16 [39].

Datasets. We evaluate our method on three datasets: (1) CIFAR-10 [20] – This dataset consists of 10 classes, with 6,000 images per class. (2) CIFAR-100 Subset – The CIFAR-100 dataset [20] consists of 100 classes with ~600 images per class. For our experiments, we selected a subset of 30 classes. (3) ImageNet Subset (ImageNette) [15] – This predefined subset of the ImageNet dataset[8] contains 10 classes with ~1,000 images each.

Figure 3: **Example of a Static Trigger Embedded into an Image.** The images demonstrate embedding a static trigger (yellow rubber duck) into an image with varying blending factors. The blending factors range from 0 (no blending) in the top left image to 1.0 (full intensity) in the bottom right image. The blending factors for the images, arranged from left to right and top to bottom, are 0, 0.2, 0.4, 0.6, 0.8, and 1.0.

Statistical Tests for the Trigger Pattern Activation Test (TPAT). For static triggers, we use the KS test [28]. The KS test is suitable for static triggers, as it is sensitive to differences in the location and shape of the empirical cumulative distribution functions of the two samples. For dynamic triggers, we employ the chi-squared test [33]. The chi-squared test is appropriate for dynamic triggers, since it can handle categorical data and assess the statistical significance of the observed differences in label distributions.

Trigger Design and Implementation. We utilize state-of-the-art generative models to create the trigger patterns used in our experiments. For static triggers, we use GPT-4 Vision (preview) [30] to generate various images, each containing a single simple object that occupies no more than 10% of the image pixels. These images are the basis for our static trigger patterns, ensuring that the triggers are visually coherent and semantically meaningful. We produced the dynamic triggers through an automatic pipeline that consists of (1) identifying the best area for applying the dynamic object (mask), using GPT-4 Vision (preview) [30], and (2) blending the object in the images (in the chosen mask area) through an inpainting process, using the DALL-E 2 API [35]. Unless mentioned otherwise, the experiments were conducted using static triggers.

Implementation Details We trained the models using the Adam optimizer [18] with a learning rate of 0.001 and a batch size of 128 for 200 epochs. We conducted experiments with various marking rates (γ), and found that 0.2% is the lowest marking rate yielding good watermarking results while maintaining high accuracy. Therefore we set $\gamma = 0.2$. The threshold for the statistical tests in TPAT and TCIT is set to 0.03.

4.2 Evaluation Metrics

Metrics for Dataset Watermarking. We adopted the metrics of benign accuracy (BA) and watermark success rate (WSR) to assess the performance of DataDetective's [24].

The benign accuracy is defined as the model's accuracy on the benign testing set, while the watermark success rate is defined as the model's accuracy on the watermarked testing set. The higher the BA and WSR values, the better the performance.

Metrics for Dataset Verification. The metrics used to evaluate the performance of the identification process are divided into two categories: (1) metrics used to evaluate individual triggers (as standalone triggers), and (2) metrics used to evaluate triggers group (signature).

Table 1: **Watermarking effectiveness results**. The table presents the average BA and WSR for different signature sizes on the CIFAR-10, CIFAR-100, and ImageNet datasets using ResNet and VGG models.

Dataset	Signature Size–>	Clean (0)	5		10		15	
	Model, Metric–>	BA	BA	WSR	BA	WSR	BA	WSR
CIFAR-10	ResNet	91.31	91.15	98.76	90.95	98.36	90.85	99.12
	VGG	90.69	90.57	98.03	90.67	96.87	90.75	96.72
CIFAR-100	ResNet	88.14	86.68	42.97	86.37	63.51	86.28	74.76
	VGG	84.33	84.25	47.98	83.15	49.46	82.86	56.26
ImageNet	ResNet	95.50	97.46	92.82	97.04	96.43	96.72	96.03
	VGG	99.30	98.96	64.08	99.04	81.61	98.9	87.02

Evaluation of Individual Triggers.

P-Value: The p-value is used to verify the effectiveness of dataset verification. We adopted [24] to evaluate our method in three scenarios: (1) leaked dataset, in which the agent triggers adopted in the suspicious watermarked model's training process are used; (2) benign model, in which the suspicious benign model is examined using the watermark trigger patterns; (3) independent trigger, in which the suspicious watermarked model is verified using a trigger which is different from those used in the training process.

In the last two scenarios, the model should not be regarded as trained on the protected dataset, and therefore, the higher the p-value, the better the verification performance. In the first scenario, the suspicious model is trained on the protected dataset, and therefore, the lower the p-value, the better the method.

Evaluation of Watermark Signature.

Signature Identification Ratio (SIR): The SIR metric measures the proportion of correctly identified tokens in a watermark signature compared to the signature's total size and quantifies the accuracy of identifying the unique watermark signature embedded in a dataset or model. Given a watermark signature $signature_i$ assigned to an agent A_i, let $signature_i^*$ denote the set of tokens (tp, tc) correctly identified in a suspected dataset or model. The SIR for A_i is:

$$SIR_i = \frac{|signature_i^*|}{|signature_i|}$$

where $|signature_i^*|$ represents the number of correctly identified tokens in the watermark signature, and $|signature_i|$ represents the total number of tokens in the original watermark signature.

The SIR ranges from 0 to 1, where $SIR_i = 1$ indicates that all tokens in the watermark signature have been correctly identified, suggesting the perfect identification of A_i's signature, and $SIR_i = 0$ indicates that none of the tokens in the watermark signature have been correctly identified, suggesting the complete failure in identifying A_i's signature. The SIR provides a standardized measure of the watermarking scheme's effectiveness in identifying each agent's unique signature. It enables comparison of the identification accuracy across different agents and datasets.

5 Results

Watermarking Effectiveness. Table 1 presents the results obtained on the CIFAR-10, CIFAR-100, and ImageNet datasets. As can be seen, for CIFAR-10 and ImageNet, the watermarking process appears to maintain high BA across different signature sizes (90.6% - 99%), where the BA for watermarked models decreases by less than 3% compared to benign models, suggesting that the utility of the model and dataset for legitimate purposes remains intact after watermarking. The WSR values are high as well (64.1% - 99.1%), indicating effective watermarking. The BA (82.9% - 87%) and WSR (43% - 74.76%) values obtained on the CIFAR-100 dataset are slightly lower than those obtained on the CIFAR-10 and ImageNet datasets, suggesting that the complexity of a dataset may influence the sensitivity to watermarking. More complex datasets are potentially more challenging for watermark embedding.

Verification Process.

Leaked Dataset Scenarios: As can be seen respectively in Table 2 and Table 3, low p-values for the TCIT and TPAT were obtained across all datasets and models, indicating that the models trained with the leaked dataset learned the association between the triggers and specific target classes. This signifies the successful embedding and identification of the trigger patterns used as watermarks.

Benign Model Scenario: In contrast to the results obtained in the leaked dataset scenario, in the benign model scenario, the p-values are high $(generally >= 0.5)$ for the TCIT and TPAT, indicating that benign models do not associate the triggers with any particular class.

Independent Trigger Scenarios: In most cases, the p-values for both the TPAT (Table 2) and TCIT (Table 3) are high, indicating that models do not incorrectly associate independent triggers with specific classes, reinforcing the robustness and specificity of the watermarking approach. Notably, the use of the VGG model on the CIFAR-100 dataset for the TPAT results in some lower p-values, suggesting a potential need for refinement in terms of the trigger design or statistical thresholds employed. Nevertheless, the lower VGG p-values for the TPAT, particularly on the CIFAR-100 dataset, are not considered false positives, since the corresponding p-values for the TCIT remain high. The results demonstrate the precision and reliability of the watermarking and leaker identification process, highlighting its potential for practical application while suggesting some areas for further refinement, especially regarding the behavior of different models in response to independent triggers.

Signature Evaluation: As seen in the results presented in Table 4, the SIR of watermarked models is consistently at 1.0 in most cases, with a slight drop to 0.98 and 0.99 in a few instances on the CIFAR-100 and ImageNet datasets with the VGG classification model. This indicates that the watermark signature is accurately detected in watermarked models in the vast majority of cases and demonstrates the method's reliability in pinpointing the leaking agent. Furthermore, the SIR values remain high across the various datasets and models, demonstrating our watermarking approach's wide applicability and robustness. In most dataset-model combinations, the SIR is 0 for benign models (not exposed to watermarked data). This is crucial, as it indicates a low false positive rate; our results show that benign models are not mistakenly identified as watermarked, which is important for preventing false accusations of unauthorized data use or dataset leakage.

Table 2: TPAT average p-value.

Scenario	Signature Size	CIFAR-10		CIFAR-100		ImageNet	
		ResNet	VGG	ResNet	VGG	ResNet	VGG
Leaked Dataset	5	$2.17e^{-4}$	$2.17e^{-4}$	$3.78e^{-3}$	$1.31e^{-10}$	$2.17e^{-4}$	$2.53e^{-4}$
	10	$2.17e^{-4}$	$2.17e^{-4}$	$4.47e^{-4}$	$1.15e^{-3}$	$2.17e^{-4}$	$4.3e^{-4}$
	15	$2.17e^{-4}$	$2.17e^{-4}$	$1.11e^{-8}$	$1.66e^{-4}$	$2.17e^{-4}$	$2.17e^{-4}$
Benign Model	5	0.83	0.73	0.75	0.51	0.86	0.77
	10	0.82	0.86	0.61	0.65	0.80	0.72
	15	0.83	0.76	0.58	0.77	0.92	0.70
Independent Trigger	5	0.76	0.80	0.54	0.68	0.92	0.49
	10	0.54	0.85	0.43	0.01	0.97	0.35
	15	0.37	0.46	0.37	0.02	0.88	0.28

Table 3: TCIT average p-value.

Scenario	Signature Size	CIFAR-10		CIFAR-100		ImageNet	
		ResNet	VGG	ResNet	VGG	ResNet	VGG
Leaked Dataset	5	$6.46e^{-21}$	$4.19e^{-20}$	$2.10e^{-27}$	$1.64e^{-14}$	$5.74e^{-12}$	$4.38e^{-65}$
	10	$7.08e^{-19}$	$1.98e^{-17}$	$5.51e^{-25}$	$2.32e^{-10}$	$1.20e^{-41}$	$2.03e^{-25}$
	15	$2.41e^{-21}$	$6.98e^{-17}$	$2.50e^{-12}$	$8.42e^{-11}$	$2.22e^{-65}$	$4.41e^{-43}$
Benign Model	5	0.99	0.90	0.99	0.99	0.97	0.96
	10	0.95	0.94	0.99	0.98	0.96	0.98
	15	0.95	0.91	0.98	0.99	0.97	0.99
Independent Trigger	5	0.97	0.52	1.0	1.0	0.99	0.95
	10	0.99	0.78	1.0	1.0	1.0	0.85
	15	0.90	0.66	0.99	0.90	0.99	0.87

Different UTP Set Sizes (Signature Sizes). We evaluate our method using UTP sets of different sizes. The p-values for the TPAT and TCIT in the leaked dataset scenarios remain low across signature sizes, indicating effective and consistent watermark detection. Moreover, the increase in the signature size from 5 to 15 does not significantly degrade the BA with the ResNet and VGG models, suggesting that larger signatures can be embedded with minimal impact on model utility.

Dynamic Trigger Evaluation. As seen in Table 5, the performance of dynamic triggers closely resembles that of static triggers. With a signature size of 3 (trigger patterns: bee, baseball, and flag), the watermarked models achieve a high WSR of 57%. Furthermore, the low p-values obtained in both the TPAT and TCIT statistical tests demonstrate the dynamic watermarking approach's effectiveness in detecting and identifying the presence of watermarks.

Dynamic Trigger Detectability Evaluation. To evaluate our watermarks' detectability, we conducted a small-scale experiment to determine whether our dynamic triggers are more difficult for an AI-based observer to detect than classic triggers. We presented ChatGPT-4 with a set of 45 images divided into three categories: 15 images with classic triggers (images containing a prominent white square patch as the trigger), 15 images with our dynamic triggers which were designed to blend into the image content seamlessly, and 15 clean, unmodified images. ChatGPT-4 was presented with the following prompt for each image: "Do you find anything abnormal or suspicious in any of the images? If so, elaborate."

The results of the experiment are summarized as follows:

- Classic Triggers: ChatGPT flagged 13 of the 15 images containing classic triggers as abnormal, identifying the white square patch as the suspicious element in these images.
- Dynamic Triggers: ChatGPT flagged only 1 of the 15 images with dynamic triggers as abnormal. The abnormality described was vague and did not specifically indicate the presence of a trigger.
- Clean Images: One of the 15 clean images was flagged as abnormal, although ChatGPT's description did not highlight any distinct feature suggesting the presence of a trigger.

These results indicate that our dynamic triggers are less detectable than the classic triggers. While ChatGPT-4 consistently identified the classic triggers, it struggled to detect the dynamic triggers, supporting our hypothesis that dynamic triggers are more stealthy.

6 Discussion

Large Number of Agents. For scenarios with an extremely large number of agents and constraints on the number of triggers that can be embedded in the dataset (total marking rate), we propose an optimized two-level watermarking approach:

1. **First Level:** The agents are divided into significantly smaller groups using the "regular" watermark signature described in Section 3. All agents in the same group are assigned identical watermark signatures.
2. **Second Level:** In each group, the agents are further differentiated by assigning multiple agents the same trigger but with different target classes. These triggers, which are not part of the UTP set used in the first level, are unique to each subgroup of agents in the group.

This two-level approach allows us to distinguish between agents in the same group who share identical watermark signatures. The identification process then follows a two-step approach:

1. **Step One:** Identify the group to which the leaking agent belongs by detecting the watermark signature common to the group.
2. **Step Two:** Iterate over the subgroups in the identified group, querying the suspected model with the corresponding triggers to identify the assigned target classes and pinpoint the exact leaking agent.

While this method requires more queries than the original approach due to the need to check each subgroup's unique triggers, it offers a trade-off between the number of embedded triggers and the number of queries required for leaker identification. This approach accommodates a larger number of agents while ensuring that the overall marking rate within the desired constraints.

Table 4: Combined average SIR for watermarked (left) and benign (right) models across the datasets. High SIR values indicate effective watermark detection.

	CIFAR-10		CIFAR-100		ImageNet	
Signature Size	ResNet	VGG	ResNet	VGG	ResNet	VGG
5	1.00 / 0.00	1.00 / 0.00	0.98 / 0.00	1.00 / 0.00	1.00 / 0.00	1.00 / 0.06
10	1.00 / 0.00	1.00 / 0.00	1.00 / 0.03	0.98 / 0.00	1.00 / 0.00	0.99 / 0.05
15	1.00 / 0.00	1.00 / 0.00	1.00 / 0.06	1.00 / 0.00	1.00 / 0.00	1.00 / 0.00

Table 5: Dynamic trigger watermark performance.

Model Type	BA	WSR	TPAT p-value	TCIT p-value
Benign Model	93.65	0	0.99	0.87
Watermarked Model	93.25	0.57	$2.34e^{-06}$	$2.18e^{-13}$

Trigger Detectability. It is important to note that while the triggers are seamlessly blended into the images, making them less conspicuous, the target class assigned to the watermarked samples may differ from the original class. This could potentially raise suspicion among agents who closely inspect the labeled data. However, there are several mitigating factors: (1) It is challenging to identify the specific trigger embedded in the watermarked images, as the triggers are blended naturally and contextually into the image content. This makes it difficult for agents to selectively remove or modify the triggers without significantly altering the image. (2) Target classes that are semantically related or visually similar to the original classes can be strategically choosen, reducing the likelihood that agents will detect discrepancies between the image content and the assigned label. (3) The dynamic trigger approach presented in this paper serves as a proof-of-concept, and our watermarking method can adapted and used in conjunction with other watermarking techniques that may offer improved stealth or imperceptibility.

7 Conclusion

In this paper, we proposed DataDetective, a novel black-box dataset watermarking approach that addresses a critical gap, detecting unauthorized use of protected datasets and efficiently identifying the agent responsible for the data leakage. By leveraging backdoor techniques, DataDetective embeds unique watermark signatures into each agent's version of a dataset, inducing distinct, agent-specific behaviors in models trained on the leaked dataset. This, along with black-box model querying requiring a low number of queries, enables leaker identification. The use of generative AI models for the seamless integration of triggers into images minimizes the risks of detection and removal by malicious actors, increasing DataDetective's robustness. By providing a robust solution for data breach attribution, this work promotes accountability in collaborative ML environments. In future research, we plan to extend DataDetective to domains beyond computer vision. Additionally, we aim to expand this work to enable DataDetective to determine whether authorized agents have used the datasets for unauthorized purposes or tasks beyond the intended scope.

References

[1] J. P. Bharadiya. Machine learning and ai in business intelligence: Trends and opportunities. *International Journal of Computer (IJC)*, 48(1):123–134, 2023.

[2] R. Bitton, N. Maman, I. Singh, S. Momiyama, Y. Elovici, and A. Shabtai. Evaluating the cybersecurity risk of real-world, machine learning production systems. *ACM Computing Surveys*, 55(9):1–36, 2023.

[3] P. BUKATY. *The California Consumer Privacy Act (CCPA): An implementation guide*. IT Governance Publishing, 2019. ISBN 9781787781320. URL http://www.jstor.org/stable/j.ctvjghvnn.

[4] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[5] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[6] Z. Cong, X. Luo, J. Pei, F. Zhu, and Y. Zhang. Data pricing in machine learning pipelines. *Knowledge and Information Systems*, 64(6):1417–1455, 2022.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[9] K. S. Erika McCallister, Tim Grance. Guide to protecting the confidentiality of personally identifiable information (pii) @ONLINE. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-122.pdf, 2010.

[10] European Commission. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), 2016. URL https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[11] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1563–1580, 2022.

[12] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[13] I. Gupta and A. K. Singh. A holistic view on data protection for sharing, communicating, and computing environments: Taxonomy and future directions. *arXiv preprint arXiv:2202.11965*, 2022.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] J. Howard. Imagenette. https://github.com/fastai/imagenette, 2019.

[16] H. Hu, Z. Salcic, G. Dobbie, J. Chen, L. Sun, and X. Zhang. Membership inference via backdooring. *arXiv preprint arXiv:2206.04823*, 2022.

[17] D. Jakhar and I. Kaur. Current applications of artificial intelligence for covid-19. *Dermatologic Therapy*, 33(4), 2020.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] M. Kop. *Chapter 22: Machine learning and EU data-sharing practices:*

Legal aspects of machine learning training datasets for AI systems. Edward Elgar Publishing, Cheltenham, UK, 2021. ISBN 9781788972819. doi: 10.4337/9781788972826.00028. URL https://www.elgaronline.com/view/edcoll/9781788972819/9781788972819.00028.xml.

[20] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[21] Y. Li, Z. Zhang, J. Bai, B. Wu, Y. Jiang, and S.-T. Xia. Open-sourced dataset protection via backdoor watermarking. *arXiv preprint arXiv:2010.05821*, 2020.

[22] Y. Li, Y. Bai, Y. Jiang, Y. Yang, S.-T. Xia, and B. Li. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. *Advances in Neural Information Processing Systems*, 35: 13238–13250, 2022.

[23] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[24] Y. Li, M. Zhu, X. Yang, Y. Jiang, T. Wei, and S.-T. Xia. Black-box dataset ownership verification via backdoor watermarking. *IEEE Transactions on Information Forensics and Security*, 2023.

[25] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, and Z. Lin. When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

[26] G. Liu, T. Xu, X. Ma, and C. Wang. Your model trains on my data? protecting intellectual property of training data via membership fingerprint authentication. *IEEE Transactions on Information Forensics and Security*, 17:1024–1037, 2022.

[27] P. Maini, M. Yaghini, and N. Papernot. Dataset inference: Ownership resolution in machine learning. *arXiv preprint arXiv:2104.10706*, 2021.

[28] F. J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[29] A. Moubayed, M. Injadat, A. B. Nassif, H. Lutfiyya, and A. Shami. E-learning: Challenges and research opportunities using machine learning & data analytics. *IEEE Access*, 6:39117–39138, 2018.

[30] OpenAI. Gpt-4 vision (preview). https://openai.com/gpt-4-vision-preview, 2023.

[31] P. Papadimitriou and H. Garcia-Molina. Data leakage detection. *IEEE Transactions on knowledge and data engineering*, 23(1):51–63, 2010.

[32] A. Paullada, I. D. Raji, E. M. Bender, E. Denton, and A. Hanna. Data and its (dis) contents: A survey of dataset development and use in machine learning research. *Patterns*, 2(11), 2021.

[33] K. Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[34] N. Peiffer-Smadja, R. Maatoug, F.-X. Lescure, E. D'ortenzio, J. Pineau, and J.-R. King. Machine learning for covid-19 needs global collaboration and data-sharing. *Nature Machine Intelligence*, 2(6):293–294, 2020.

[35] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[36] A. Sablayrolles, M. Douze, C. Schmid, and H. Jégou. Radioactive data: tracing through training. In *International Conference on Machine Learning*, pages 8326–8335. PMLR, 2020.

[37] A. Saha, A. Subramanya, and H. Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11957–11965, 2020.

[38] A. Schwarzschild, M. Goldblum, A. Gupta, J. P. Dickerson, and T. Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *International Conference on Machine Learning*, pages 9389–9398. PMLR, 2021.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] C. Song and V. Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206, 2019.

[41] R. Tang, Q. Feng, N. Liu, F. Yang, and X. Hu. Did you train on my dataset? towards public dataset protection with clean-label backdoor watermarking. *arXiv preprint arXiv:2303.11470*, 2023.

[42] E. Wenger, X. Li, B. Y. Zhao, and V. Shmatikov. Data isotopes for data provenance in dnns. *arXiv preprint arXiv:2208.13893*, 2022.

[43] H. Yan, X. Li, H. Li, J. Li, W. Sun, and F. Li. Monitoring-based differential privacy mechanism against query flooding-based model extraction attack. *IEEE Transactions on Dependable and Secure Computing*, 19 (4):2680–2694, 2021.

[44] Z. Zou, B. Gong, and L. Wang. Anti-neuron watermarking: Protecting personal data against unauthorized neural networks. In *European Conference on Computer Vision*, pages 449–465. Springer, 2022.